УСТАНОВКА APACHE И PHP

Чтобы работать со скриптами php, необходим HTTP сервер и интерпретатор php. Ниже приведена инструкция по установке apache и php.

Установка Apache2.2.2:

Загрузите с сайта www.apache.org дистрибутив Apache2.2.2 в виде инсталлятора с именем apache_2.2.2-win32-x86-no_ssl.msi. Вы сможете найти его по адресу www.sai.msu.su/apache/dist/httpd/binaries/win32/.

Запустите загруженный инсталлятор на выполнение. Когда установщик спросит Вас о том, куда установить Apache укажите ему директорию c:/Apache2.2

Примечание

Все дальнейшие инструкции будут основываться на предположении, что Apache устанавливается именно в каталог c:/Apache2.2. Если Вы устанавливаете Apache в другой каталог, то Вы должны соответствующим образом адаптировать инструкции к своей ситуации.

Установка Apache из инсталлятора достаточно прозрачна и не вызывает особых трудностей, вследствие чего не рационально приводить ее полное описание. Приведем лишь одно диалоговое окно, которые пользователю требуется заполнить в процессе установки. Это окно выбора имени сервера. В поля «Network Domain» и «ServerName» запишите имя сервера, на работу с которым Apache будет настроен по умолчанию.

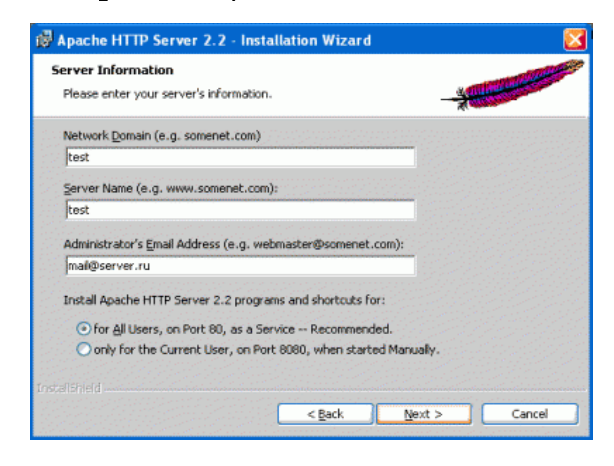


Рис. 1 Окно установки Apache

Примечание

Данная инструкция описывает установку сервера Apache в предположении, что он будет использоваться только для локального тестирования сайтов и не будет работать в сетях Интранет и Интернет. Для работы сервера в сетях Интранет и Интернет Вы должны ввести реальное доменное имя, которые будет использоваться сервером.

Если процесс установки прошел корректно, то по ее завершению Apache2.2 у Вас уже должен быть запущен в качестве службы. Чтобы проверить так ли это откройте список служб Windows («Пуск» | «Панель управления» | «Администрирование» | «Службы») и найди в нем строку Apache2.2. (или Apache2). Состояние службы: работает или нет отображается в третьем столбце.

Если Вы не можете найти в списке служб строчку Apache2.2, то вероятно в процессе установки произошел сбой и Apache не установился в качестве службы. В этом случае, Вам необходимо установить Apache, в качестве службы самостоятельно. Для выполнения этого понадобится программа с консолью, например FAR, WindowsCommander, TotalCommander и т.п.

Откройте программу с консолью, перейдите в каталог c:/Apache2.2/bin и выполните команду:

```
C:/Apache2.2/bin/httpd.exe -k install
```

В ответ Вам должно быть выдано сообщение «The Apache2 service is successfully installed». Внешний вид консольного окна приведен на рис. 2

```
C:\Apache2.2\bin>httpd.exe -k install
Installing the Apache2 service
The Apache2 service is successfully installed.
```

Рис. 2 Вид консольного окна

Первый запуск: Управление Apache-ем (пуск, остановка, перезапуск) осуществляется либо через графический интерфейс управления службами Windows, либо в консоли выполнением файла httpd.exe с определенными ключами.

Управление службами Windows осуществляется через контекстное меню, открываемое по нажатию правой кнопки мыши на имени службы. Если Apache еще не запущен, то выполните команду «Пуск» из контекстного меню.

Если Вам нравится работать в консоли то для управления Apache используются ключи, приведенные в табл 1.

Команды управления Apache через консоль	
httpd.exe -k start	Запуск службы
httpd.exe -k stop	Остановка службы
httpd.exe -k restart	Перезапуск

Табл. 1 Команды управления Apache через консоль

Далее проверьте работу сервера через браузер. Наберите в своем любимом браузере имя localhost в адресной строке. В ответ Вам должна быть выдана страница, с текстом «It works!».

Добавление виртуальных хостов: чтобы добавить виртуальных хост к вашему серверу, необходимо открыть файл C:/Apache2.2/conf/httpd.conf, найти строку `DocumentRoot` и указать путь к папке, где вы собираете хранить файлы для хостов. Например

```
DocumentRoot "C:/Sites/home/localhost/www"
```

Теперь создайте каталог `C:/Sites/home/localhost` и в нем подкаталоги cgi, www, а так же файлы access.log и error.log. Эта процедура выполняется единожды. При добавлении хоста нужно произвести следующие действия. Создать папку в каталоге C:/Sites/home с именем вашего нового хоста. Внутри нее создать подкаталоги cgi, www, а так же файлы access.log и error.log. В файле C:/Apache2.2/conf/httpd.conf добавить хост

```
<VirtualHost phpMyAdmin>
  ServerAdmin me@phpMyAdmin
  ServerName phpMyAdmin.com
  DocumentRoot "C:/Sites/home/phpMyAdmin/www"
```

```
  ScriptAlias /cgi/ "C:/Sites/home/phpMyAdmin/cgi/"
  ErrorLog C:/Sites/home/phpMyAdmin/error.log
  CustomLog C:/Sites/home/phpMyAdmin/access.log common
</VirtualHost>
```

В файле C:/Windows/System32/drivers/etc/hosts прописать следующую строку

```
127.0.0.1 phpMyAdmin.com
```

Перезапустите Apache и из браузера можете обращаться к вашему новому виртуальному хосту по адресу phpMyAdmin.com.

Установка PHP: современный web-сервер уже немыслим без поддержки динамически-генерируемых страниц. В Украине лидером среди технологий создания динамических страниц и web-приложений является технология PHP. Ниже будет рассмотрен процесс подключения PHP на примере версии 5.3.5. Если у Вас уже установлена эта версия PHP, то можете сразу переходить к разделу «Подключение PHP к Apache2.2.2».

Дистрибутив с PHP Вы можете загрузить с официального сайта по адресу http://windows.php.net/download/. При загрузке дистрибутива, представленного по ссылке, предполагается, что вы будете устанавливать PHP в качестве модуля.

Распакуйте zip-архив с php в директорию c:/php-5.3.5. На этом установка PHP завершена. Дальнейшие действия будут касаться настройки связки PHP+Apache и конфигурирования самого PHP.

Для подключения php, в качестве модуля, необходимо добавить всего 3 инструкции в файл httpd.conf.

Инструкции для подключения PHP в httpd.conf
LoadModule php5_module c:/php-5.3.5/php5apache2_2.dll AddType application/x-httpd-php phtml php PHPIniDir "c:/php-5.3.5/"

Табл. 2 Инструкции для подключения PHP

Эти строки следует поместить в примерно в середину файла httpd.conf, например, сразу после директив LoadModule. Конкретное расположение этих директив не имеет принципиального значения, однако нельзя помещать их в начале, а также и в самом конце файла httpd.conf.

Первая строка загружает модуль PHP, реализованный в библиотеке php5apache2_2.dll. Вторая строка устанавливает соответствие между файлами с расширением php и mime-типом application/x-httpd-php, который обрабатывается модулем PHP.Третья строка позволяет явно указать расположением конфигурационного файла php.ini.

Следующим шагом необходимо создать конфигурационный файл для PHP. В httpd.conf в директиве PHPIniDir местом расположения конфигурационного файла php была указана директория c:/php-5.3.5. Сам конфигурационный файл должен называться php.ini

В директории c:/php-5.3.5 находятся несколько шаблонов конфигурационных файлов. В качестве основы возьмем файл c:/php-5.3.5/php.ini-recommended и переименуем его в php.ini. Таким образом, конфигурационный файл PHP (php.ini) будет располагаться в директории c:/php-5.3.5 и именно в него должны вносится все изменения конфигурации PHP.

После внесения изменений в httpd.conf и создания файла php.ini перезагрузите Apache.

Создайте тестовый php-скрипт под именем phpinfo.php, выполняющий одноименную функцию и сохраните его в директорию c:/Apache2.2/htdocs.

```
Скрипт phpinfo.php
<?php
  echo phpinfo();
?>
```

Теперь обратитесь к данному скрипту через браузер введя в адресной строке http://localhost/phpinfo.php. В ответ Вам должны быть отображены широко известные фиолетовые страницы, отображающие настройки php и его расширений.



Рис. 3 Проверка работы PHP

PHP КУКИ (COOKIES)

Cookies – это механизм хранения данных браузером удаленного компьютера для идентификации возвращающихся посетителей и хранения параметров веб-страниц (например, переменных).

Приведем пример использования Cookies на конкретном примере.

Предположим, нам нужно написать счетчик посещения сайта. Нам нужно знать, какое число посещений сайта осуществлялось каждым конкретным посетителем.

Данную задачу можно решить двумя способами. Первый из них заключается в ведении учета IP-адресов пользователей. Для этого нужна база данных всего из одной таблицы, примерная структура которой такая:

IP-адрес	Число посещений
210.124.134.203	7
212.201.78.207	14
83.103.203.73	3

Когда пользователь заходит на сайт, нам нужно определить его IP-адрес, найти в базе данных информацию о его посещениях, увеличить счетчик и вывести его в браузер посетителя. Написать обработчик (скрипт) подобной

процедуры несложно. Однако при использовании такого метода у нас появляются проблемы следующего характера:

- Для каждого IP-адреса нужно вести учет в одной таблице, которая может быть очень большой. А из этого следует, что мы нерационально используем процессорное время и дисковое пространство;
- У большинства домашних пользователей IP-адреса являются динамическими. То есть, сегодня у него адрес 212.218.78.124, а завтра - 212.218.78.137. Таким образом, велика вероятность идентифицировать одного пользователя несколько раз.

Можно использовать второй способ, который намного легче в реализации и более эффективен. Мы устанавливаем в Cookie переменную, которая будет храниться на диске удаленного пользователя. Эта переменная и будет хранить информацию о посещениях. Она будет считываться скриптом при обращении посетителя к серверу. Выгода такого метода идентификации очевидна. Во-первых, нам не нужно хранить множество ненужной информации о IP-адресах. Во-вторых, нас не интересуют динамические IP-адреса, поскольку данные о своих посещениях хранятся конкретно у каждого посетителя сайта.

Теперь понятно, для чего мы можем использовать Cookie −для хранения небольшой по объему информации у клиента (посетителя) сайта, например: настройки сайта (цвет фона страниц, язык, оформление таблиц и т.д.), а также другой информации.

Файлы Cookies представляют собой обыкновенные текстовые файлы, которые хранятся на диске у посетителей сайтов. Файлы Cookies и содержат ту информацию, которая была в них записана сервером.

Программирование Cookies

Приступим к программированию Cookies.

Для установки Cookies используется функция SetCookie(). Для этой функции можно указать шесть параметров, один из которых является обязательным:

- name - задает имя (строк), закрепленное за Cookie;
- value - определяет значение переменной (строка);
- expire - время "жизни" переменной (целое число). Если данный параметр не указать, то Cookie будут "жить" до конца сессии, то есть

до закрытия браузера. Если время указано, то, когда оно наступит, Cookie самоуничтожится.

- path - путь к Cookie (строка);
- domain - домен (строка). В качестве значения устанавливается имя хоста, с которого Cookie был установлен;
- secure - передача Cookie через защищенное HTTPS-соединение.

Обычно используются только три первые параметра.

Пример установки Cookies:

```
<?php
  // Устанавливаем Cookie до конца сессии:
  SetCookie("Test","Value");
  // Устанавливаем Cookie на один час после установки:
  SetCookie("My_Cookie","Value",time()+3600);
?>
```

При использовании Cookies необходимо иметь в виду, что Cookies должны устанавливаться до первого вывода информации в браузер (например, оперетором echo или выводом какой-либо функции). Поэтому желательно устанавливать Cookies в самом начале скрипта. Cookies устанавливаются с помощью определенного заголовка сервера, а если скрипт выводит что-либо, то это означает, что начинается тело документа. В результате Cookies не будут установлены и может быть выведено предупреждение. Для проверки успешности установки Cookies можно использовать такой метод:

```
<?php
  // Устанавливаем Cookie до конца сессии:
  // В случае успешной установки Cookie, функция SetCookie
  //возвращает TRUE:
  if (SetCookie("Test","Value"))
       echo "<h3>Cookies успешно установлены!</h3>";
?>
```

Функция SetCookie() возвращает TRUE в случае успешной установки Cookie. В случае, если Cookie установить не удается SetCookie() возвратит FALSE и возможно, предупреждение (зависит от настроек PHP). Пример неудачной установки Cookie:

```
<?php
```

```
  // Cookies установить не удастся, поскольку перед отправкой
  // заголовка Cookie мы выводим в браузер строку 'Hello':
  echo "Hello";
  // Функция SetCookie возвратит FALSE:
  if (SetCookie("Test","Value"))
       echo "<h3>Cookie успешно установлен!</h3>";
  else
       echo "<h3>Cookie установить не удалось!</h3>";
  // Выводит 'Cookie установить не удалось!'.
?>
```

Cookie установить не удалось, поскольку перед посылкой заголовка Cookie мы вывели в браузер строку "Hello".

Чтение значений Cookies

Получить доступ к Cookies и их значениям достаточно просто. Они хранятся в суперглобальных массивах и $_COOKIE и $HTTP_COOKIE_VARS.

Доступ к значениям осуществляется по имени установленных Cookies, например:

```
echo $_COOKIE['my_cookie'];
// Выводит значения установленной Cookie 'My_Cookie'
```

Пример установки Cookie и последующего его чтения:

```
<?php
  // Устанавливаем Cookie 'test' со значением 'Hello' на один
  //час:
  setcookie("test","Hello",time()+3600);
  // При следующем запросе скрипта выводит 'Hello':
  echo @$_COOKIE['test'];
?>
```

В рассмотренном примере при первом обращении к скрипту устанавливается Cookie "test" со значением "hello". При повторном обращении к скрипту будет выведено значение Cookie "test", то есть строка "Hello".

При чтении значений Cookies обращайте внимание на проверку существования Cookies, например, используя оператор isset(). Либо путем подавления вывода ошибок опереатором @

А вот пример, как построить счетчик числа загрузок страницы с помощью Cookies:

```
<?php
  // Проверяем, был ли уже установлен Cookie 'Mortal',
  // Если да, то читаем его значение,
  // И увеличиваем значение счетчика обращений к странице:
  if (isset($_COOKIE['Mortal']))
       $cnt=$_COOKIE['Mortal']+1;
  else
       $cnt=0;
  // Устанавливаем Cookie 'Mortal' зо значением счетчика,
  // С временем "жизни" до 18/07/29,
  // То есть на очень долгое время:
  setcookie("Mortal",$cnt,0x6FFFFFFF);
  // Выводит число посещений (загрузок) этой страницы:
  echo "<p>Вы посещали эту страницу <b>".@$_COOKIE['Mortal'].
  "</b> раз</p>";
?>
```

Удаление Cookies

Иногда возникает необходимость удаления Cookies. Сделать это несложно, необходимо лишь вновь установить Cookie с идентичным именем и пустым параметром. Например:

```
<?php
  // Удаляем Cookie 'Test':
  SetCookie("Test","");
?>
```

Установка массива Cookies и его чтение

Мы может установить массив Cookies, используя квадратные скобки в именах Cookies [], а затем прочитать массив Cookies и значения этого массива:

```
<?php
  // Устанавливаем массив Cookies:
  setcookie("cookie[1]", "Первый");
  setcookie("cookie[2]", "Второй");
  setcookie("cookie[3]", "Третий");

  // После перезагрузки страницы мы отобразим
  // Состав массива Cookies 'cookie':
  if (isset($_COOKIE['cookie'])) {
       foreach ($_COOKIE['cookie'] as $name => $value) {
            echo "$name : $value <br>";
```

```
            }
    }
?>
```

Преимущества использования Cookies неоспоримы. Однако существуют и некоторые проблемы их использования. Первая из них заключается в том, что посетитель может блокировать прием Cookies браузером или попросту удалить все Cookies или их часть. Таким образом, мы можем иметь некоторые проблемы в идентификации таких посетителей.

PHP СЕССИИ(SESSIONS)

Введение

Сессии − это на самом деле очень просто. Надо только понимать, для чего они нужны и как устроены. Ответим сначала на первый вопрос.

Возможно Вы знаете, что веб-сервер не поддерживает постоянного соединения с клиентом, и каждый запрос обрабатывается, как новый, без связи с предыдущими.

То есть, нельзя ни отследить запросы от одного и того же посетителя, ни сохранить для него переменные между просмотрами отдельных страниц. Вот для решения этих двух задач и были изобретены сессии.

Собственно, сессии, если в двух словах − это механизм, позволяющий однозначно идентифицировать браузер и создающий для этого браузера файл на сервере, в котором хранятся переменные сеанса.

Подробно расписывать нужду в таком механизме я не буду. Это такие случаи, как корзина покупок в интернет магазине, авторизация, а так же, и не совсем тривиальные проблемы, такие, например, как защита интерактивных частей сайта от спама.

В принципе, довольно несложно сделать собственный аналог сессий, не такой функциональный, как встроенный в PHP, но похожий по сути. На cookies и базе данных.

При запросе скрипта смотрим, пришла ли cookies с определенным именем. Если cookies нет, то ставим ее и записываем в базу новую строку с данными пользователя. Если cookies есть, то читаем из базы данные. Еще одним запросом удаляем из базы старые записи и вот у нас готов механизм сессий. Совсем несложно. Но есть некоторые нюансы, которые делают предпочтительным использование именно встроенного механизма сессий.

Как устроены, и как работают сессии?

Для начала надо как-то идентифицировать браузер. Для этого надо выдать ему уникальный идентификатор и попросить передавать его с каждым запросом.

Сессии используют стандартные, хорошо известные способы передачи данных. Собственно, других-то просто и нет.

Идентификатор − это обычная переменная. По умолчанию ее имя − PHPSESSID.

Задача PHP отправить ее браузеру, чтобы тот вернул ее со следующим запросом. Ясно, что переменную можно передать только двумя способами: в cookies или POST/GET запросом.

PHP использует оба варианта.

За это отвечают две настройки в php.ini:

session.use_cookies− если равно 1, то PHP передает идентификатор в cookies, если 0 - то нет.

session.use_trans_sid если равно 1, то PHP передает его, добавляя к URL и формам, если 0 - то нет.

Менять эти и другие параметры сессий можно так же, как и другие настройки PHP − в файле php.ini, а так же с помощью команды ini_set() или в файлах настройки веб-сервера

Если включена только первая, то при старте сессии (при каждом вызове session_start()) клиенту устанавливается cookies. Браузер исправно при каждом следующем запросе эту cookies возвращает и PHP имеет идентификатор сессии. Проблемы начинаются, если браузер cookies не возвращает. В этом случае, не получая cookies с идентификатором, PHP будет все время стартовать новую сессию, и механизм работать не будет.

Если включена только вторая, то cookies не выставляется. А происходит то, ради чего, в основном, собственно, и стоит использовать встроенный механизм сессий. После того, как скрипт выполняет свою работу, и страница полностью сформирована, PHP просматривает ее всю и дописывает к каждой ссылке и к каждой форме передачу идентификатора сессии. Это выглядит примерно так:

```
<a href="/index.php">Index</a>
```

превращается в

```
<a href="/index.php?PHPSESSID=9ebca8bd62c830d3e9274f585ff8f">
Index</a>
```

а к формам добавляется скрытое поле

```
<input type="hidden" name="PHPSESSID"
vaue="00196c1c1a02e4c37ac04f921f4a5eec"/>
```

И браузер при клике на любую ссылку, или при нажатии на кнопку в форме, пошлет в запросе нужную нам переменную − идентификатор сессии!

Теоретически, в наших с вами самодельных сессиях на cookies и базе, можно самому, руками приписать ко всем ссылками передачу ид −и тогда наши собственные сессии будут работать независимо от cookies. Но, согласитесь− приятнее, когда эту работу делает кто-то другой?

По умолчанию в последних версиях PHP включены обе опции. Как PHP поступает в этом случае? Кука выставляется всегда. А ссылки автодополняются только если PHP не обнаружил cookies с идентификатором сессии. Когда пользователь в первый раз за этот сеанс заходит на сайт, ему ставится cookies, и дополняются ссылки. При следующем запросе, если cookies поддерживаются, PHP видит cookies и перестает дополнять ссылки. Если cookies не работают, то PHP продолжает исправно добавлять ид к ссылкам, и сессия не теряется.

Пользователи, у которых работают cookies, увидят длинную ссылку с ID только один раз.

С передачей идентификатора закончили. Теперь осталось привязать к нему файл с данными на стороне сервера. PHP это сделает за нас. Достаточно просто написать:

```
session_start();
$_SESSION['test']='Hello world!';
```

И PHP запишет в файл, связанный с этой сессией, переменную test.

Здесь очень важное замечание.

Массив $_SESSION − особенный.

В нем, собственно, и находятся переменные, которые мы ходим сделать доступными в различных скриптах.

Чтобы поместить переменную в сессию, достаточно присвоить ее элементу массива $_SESSION.

Чтобы получить ее значение − достаточно обратиться к тому же элементу. Пример будет чуть ниже.

Сборкой мусора − удалением устаревших файлов PHP тоже занимается сам. Как и кодированием данных и кучей всяких других нужных вещей. В результате этой заботы работа с сессиями оказывается очень простой.

Вот мы, собственно, и подошли к примеру работы сессий.

Пример очень маленький:

```
<?
  session_start();
  if (!isset($_SESSION['counter']))
       $_SESSION['counter']=0;
  echo "Вы обновили эту страницу ".$_SESSION['counter']++."
раз. ";
  echo "<br><a href=".$_SERVER['PHP_SELF'].">обновить</a>";
?>
```

Мы проверяем, есть ли у нас в сессии переменная counter, если нет, то создаем ее со значением 0, а дальше выводим ее значение и увеличиваем на единицу. Увеличенное значение запишется в сессию, и при следующем вызове скрипта переменная будет иметь значение 1, и так далее. Все очень просто.

Для того, чтобы иметь доступ к переменным сессии на любых страницах сайта, надо написать ТОЛЬКО ОДНУ(!) строчку в самом начале КАЖДОГО файла, в котором нам нужны сессии:

session_start();

И далее обращаться к элементам массива $_SESSION. Например, проверка авторизации будет выглядеть примерно так:

```
session_start();
if ($_SESSION['authorized']<>1) {
  header("Location: /auth.php");
  exit;
}
```

Удаление переменных из сессии. Если у вас register_globals=off, то достаточно написать

unset($_SESSION['var']);

Если же нет, то тогда рядом с ней надо написать:

session_unregister('var');

Область применения

Очень важно понимать, для чего сессии стоит использовать, а для чего − нет.

Во-первых, помните, что сессии можно применять только тогда, когда они нужны самому пользователю, а не для того, чтобы чинить ему препятствия. Ведь он в любой момент может избавиться от идентификатора!

Скажем, при проверке на то, что заполняет форму человек, а не скрипт, пользователь сам заинтересован в том, чтобы сессия работала - иначе он не

сможет отправить форму! А вот для ограничения количества запросов к скрипту сессия уже не годится - злонамеренный скрипт просто не будет возвращать идентификатор.

Во-вторых. Важно четко себе представлять тот факт, что сессия − это сеанс работы с сайтом, так как его понимает человек. Пришел, поработал, закрыл браузер − сессия завершилась. Как сеанс в кино. Хочешь посмотреть еще один – покупай новый билет. Стартуй новый сеанс. Этому есть и техническое объяснение. Гарантированно механизм сессий работает только именно до закрытия браузера. Ведь у клиента могут не работать cookies, а в этом случае, естественно, все дополненные идентификатором ссылки пропадут с его закрытием.

Правда, сессия может пропасть и без закрытия браузера. В силу ограничений, рассмотренных в этой статье, механизм сессий не может определить тот момент, когда пользователь закрыл браузер. Для этого используется таймаут – заранее определенное время, по истечении которого мы считаем, что пользователь ушел с сайта. По умолчанию этот параметр равен 24 минутам.

Если вы хотите сохранять пользовательскую информацию на более длительный срок, то используйте cookies и, если надо − базу данных на сервере. В частности, именно так работают все популярные системы авторизации:

- по факту идентификации пользователя стартует сессия и признак авторизованности передается в ней.
- Если надо "запомнить" пользователя, то ему ставится cookies, его идентифицирующая.
- При следующем заходе пользователя на сайт, для того, чтобы авторизоваться, он должен либо ввести пароль, либо система сама его опознает по поставленной ранее cookies, и стартует сессию. Новую сессию, а не продолжая старую.

В-третьих, не стоит стартовать сессии без разбору, каждому входящему на сайт. Это создаст совершенно лишнюю нагрузку. Не используйте сессии по пустякам – к примеру, в счетчиках. То, что спайлог называет сессиями, считается, конечно же, на основе статистики заходов, а не с помощью механизма сессий, аналогичного PHP.

К тому же, возьмем поисковик, который индексирует ваш сайт. Если поисковый робот не поддерживает cookies, то PHP по умолчанию будет постав-

лять к ссылкам PHPSESSID, что может не сильно понравится поисковику, который, по слухам, и так-то динамические ссылки не жалует, а тут вообще при каждом заходе – новый адрес!

Если сессии используются для ограничения доступа к закрытому разделу сайта, то все просто поисковик и не должен его индексировать. Если же приходится показывать одну и ту же страницу как авторизованным, так и не авторизованным пользователям, то тут поможет такой трюк – стартовать сессию только тем, кто ввел пароль, или тем, у кого уже стартовала сессия.

Для этого в начало каждой страницы вместо просто session_start() пишем:

```
if (isset($_REQUEST[session_name()])) session_start();
```

таким образом, Мы стартуем сессию только тем, кто прислал идентификатор.

Соответственно, надо еще в первый раз отправить его пользователю – в момент авторизации.

Если имя и проль верные – пишем session_start()

Возможные проблемы и их устранение

Самыми распространенными ошибками, которые выдает PHP при попытке работать с сессиями, являются такие:

Две из них,

Warning: Cannot send session cookie - headers already sent

Warning: Cannot send session cache limiter - headers already sent

вызваны одной и той же причиной, решение описано в этом факе здесь

Третья,

Warning: open(/tmp\sess_SID, O_RDWR) failed: No such file or directory (2) in full_script_path on line number

ранее она выглядела, как

Warning: Failed to write session data (files). Please verify that the current setting of session.save_path is correct (/tmp),

если перевести ее с английского, подробно объясняет проблему: недоступен указанный в php.ini путь к каталогу, в который пишутся файлы сессий.

Эту ошибку исправить проще всего. Просто прописать каталог, который существует, и доступен на запись, например,

session.save_path = c:\windows\temp

И не забыть перезагрузить Apache после этого.

Как выясняется, сообразительность людская не имеет пределов, и поэтому я вынужден пояснить:

сообщение о третьей ошибке (невозможно найти каталог) НЕИЗБЕЖНО приведет к появлению первых двух, поскольку сообщение об ошибке - это вывод в браузер и после него заголовками пользоваться нельзя. Поэтому не спешите искать преждевременный вывод, а сначала пропишите правильный путь!

Следующей по распространенности проблемой при работе с сессиями является тяжелое наследие register_globals. Не давайте переменным скрипта имена, совпадающие с индексами массива $_SESSION!

При register_globals=on значения будут перезаписывать друг друга, и вы запутаетесь.

Если не работает, но и никаких сообщений не выводится, то добавьте в самое начало скрипта две строчки, отвечающие за вывод ВСЕХ ошибок на экран - вполне возможно, что ошибки есть, но вы их просто не видите.

```
ini_set('display_errors',1);
error_reporting(E_ALL);
```

или смотрите ошибки в error_log. Вообще, тема отображения сообщений об ошибках выходит за рамки данной статьи, поэтому просто убедитесь хотя бы, что вы можете их видеть. Чуть продробнее о поиске ошибок можно прочитать в этом разделе.

Если вы уверены, что ошибок нет, но приведенный пример не работает все равно, то, возможно, в PHP не включена передача ид через урл, а cookies по каким-то причинам не работают.

Смотрите, что у вас с cookies.

Вообще, если у вас "не работают" сессии, то сначала попробуйте передать идентификатор сессии руками, то есть, сделать ссылку и приписать к ней идентификатор:

```
<?
  session_start();
  if (!isset($_SESSION['counter']))
```

```
        $_SESSION['counter']=0;
  echo "Вы обновили эту страницу ".$_SESSION['counter']++." раз.<br>
  <a href=".$_SERVER['PHP_SELF'].'?'.session_name().'='.session_id().">обновить</a>";
?>
```

При этом следует убедится, что не включена директива session.use_only_cookies, которая запрещает PHP принимать идентификатор сессии, если он был передан через URL

Если этот пример не заработает, то проблема либо в банальных опечатках (половина "проблем" с сессиями происходит от неправильно написанного имени переменной), либо в слишком старой версии PHP: поддержка сессий появилась в версии 4.0, а массив $_SESSION - в 4.1 (До этого использовался $HTTP_SESSION_VARS).

Если же заработает - то проблема в cookies. Отслеживайте - что за cookies ставит сервер браузеру, возвращает ли браузер ее. Искать очень полезно, просматривая обмен HTTP-заголовками между браузером и сервером.

Объяснение принципа работы cookies выходит за рамки этого и так уж слишком большого текста, но хотя бы убедитесь, что сервер cookies с идентификатором посылает, а браузер - возвращает. И при этом идентификаторы совпадают друг с другом.

Установка cookies должна выглядеть, как

```
Set-Cookie: PHPSESSID=prlgdfbvlg5fbsbshch6hj0cq6;
```

или как

```
Set-Cookie: PHPSESSID=prlgdfbvlg5fbsbshch6hj0cq6; path=/
```

(если вы запрашиваете скрипт не из корневого каталога)

Ответ сервера должен выглядеть, как

```
Cookie: PHPSESSID=prlgdfbvlg5fbsbshch6hj0cq6
```

либо

```
Cookie: PHPSESSID=prlgdfbvlg5fbsbshch6hj0cq6; b=b
```

если браузер возвращает другие cookies, кроме идентификатора сессии.

Если пример отсюда работает, а ваш собственный код - нет, то проблема, очевидно, не в сессиях, а в алгоритме. Ищите, где потеряли переменную, по шагам переносите пример отсюда, отлаживайте свой скрипт.

Еще одна проблема может возникнуть, если вы используете перенаправление через header или навигацию с помощью JavaScript.

Дело в том, что PHP автоматически дописывает идентификатор сессии только к ссылкам вида <a href=>, но не делает этого для header-ов, яваскрипта, мета-тегов.

Поэтому надо добавлять идентификатор руками, например, так:

```
header("Location:/script.php?".session_name().'='. session_id());
```

Так же, весьма редкая, и совершенно непонятно, откуда появляющаяся, проблема бывает в том, что настройка session.save_handler имеет значение, отличное от files. Если это не так - исправляйте.

Дополнительная информация:

Кроме cookies, механизм сессий посылает еще и заголовки, запрещающие кэширование страниц (тот самый cache limiter). Для html это правильно и необходимо. Но вот когда вы пытаетесь скриптом, проверяющим авторизацию, отдать файл, то интернет эксплорер отказывается его скачивать. Именно из-за этого заголовка. Вызов

session_cache_limiter("private");

перед стартом сессии должен решить проблему.

Как это ни кажется странным, но в массиве $_SESSION нельзя использовать числовые индексы - $_SESSION[1], $_SESSION['10'] - сессии работать не будут.

Где-то между версиями 4.2 и 5.0 невозможно было установить session.use_trans_sid с помощью ini_set(). Начиная с 5.0 уже можно снова.

До версии 4.3.3 cookies PHP отправлял cookies только если при старте сессии в запросе отсутствовал идентификатор. Теперь же cookies посылается при каждом вызове session_start()